Third-Party Pixel Service

Third-Party Pixel Service

The third-party pixel service allows you to upload and modify third-party creative pixels, and attach them to advertisers, or creatives. You can also attach third-party creative pixels to your member.

On This Page

REST API

Add a new third-party pixel:

```
POST https://api.appnexus.com/thirdparty-pixel
```

Modify an existing third-party pixel:

```
PUT https://api.appnexus.com/thirdparty-pixel?id=THIRDPARTY-PIXEL_ID
```

View all of your third-party pixels:

```
GET https://api.appnexus.com/thirdparty-pixel
```

View a specific third-party pixel:

```
GET https://api.appnexus.com/thirdparty-pixel?id=THIRDPARTY-PIXEL_ID
```

Delete an existing third-party pixel:

```
DELETE https://api.appnexus.com/thirdparty-pixel?id=THIRDPARTY-PIXEL_ID
```

JSON Fields

Field	Type	Description	Default	Require
id	int	The ID of the third-party pixel.		`PUT/DEL`
active	boolean	Flag indicating whether the pixel is active.	`true`	
name	string	The name of the third-party pixel.		

member_id	int	ID of the member/network that owns this pixel.		POST (sp within the string)
advertiser_id	int	ID of the advertiser that owns the thirdparty-pixel.	`null`	
format	string	The format of the pixel. Possible values for creatives: `"raw-js"`, `"url-html"`, `"url-js"`, `"url-image"`, or `"raw-url"`.	`"raw-js"`	`POST/PU` changing `t`, `secur` `nt`, `url`, `ecure` fi
content	string	If the pixel's `format` is `"raw-js"`, this is the JavaScript content to serve with the creative. The `content` and/or `secure_content` fields are required on `POST` for raw-js pixels. You can also add macros to your pixel. For a list of the creative macros that you may append to your pixel, see Creative Macros (Customer login required).		`POST` if th `format` `js"`.
secure_content	string	If the pixel's `format` is "raw-js", the JavaScript content to serve with the creative. The `content` and/or `secure_content` fields are required on POST for raw-js pixels. You can also add macros to your pixel. For a list of the creative macros that you may append to your pixel; see Creative Macros (Customer login required).		`POST` if th `format` `js"`.

url	string	If the pixel's `format` is "url-html", "url-js", "url-image", or "raw-url", the URL of the HTML, JavaScript, or Image pixel to serve with the creative. The `url` and/or `secure_url` fields are required on POST for these pixel types. You can also add macros to your pixel. For a list of the creative macros that you may append to your pixel, see Creative Macros (Customer login required).		
secure_url	string	If the pixel's `format` is "url-html", "url-js", "url-image", or "raw-url", the URL of the HTML, JavaScript, or Image pixel to serve with the creative on a secure (https) call. The url and/or secure_url fields are required on `POST` for these pixel types. You can also add macros to your pixel. You can also add macros to your pixel. For a list of the creative macros that you may append to your pixel, see Creative Macros (Customer login required).		`POST` if tl `format` `html"`, ' , `"url-j` or `"raw-`
members	array of objects	**Optional.** If specified, this field will contain the ID of the member that owns the pixel and this pixel will render on all creatives owned by that member. Example: `"members":[{"id":1}]`		

advertisers	array of objects	**Optional.** If the pixel has an owning advertiser (i.e., `advertiser_id` is not `null`), this array must contain the ID (and only the ID) of the advertiser that owns the pixel, As a result, this pixel will render on all creatives owned by that advertiser. If the pixel does not have an owning advertiser, this array can contain multiple advertisers to which the pixel will be applied. Example: `"advertisers":[{"id":3}]`		
creatives	array of objects	**Read only.** The creatives to which a pixel is applied. The creatives must belong to the owning member/advertiser. To attach a third-party pixel to a creative, use the Creative Service. Example: `"creatives": [{"id":860851}]`		
audit_status	string	**Read-only.** The audit status of the pixel. Possible values are `"pending"`, `"rejected"`, `"approved"` or `"exempt"`. An unaudited pixel will not prevent a creative from serving, but it will not serve along with the creative until it has passed audit.	`"pending"`	
supply_exceptions	array of objects	The names of members on which the pixel should **not** serve. `"AdX"` is currently the only acceptable value. Example: `"supply_exceptions":[{"name":"AdX"}]`		

adservers	array of objects	The adservers that the pixel calls. Required for pixels that serve on AdX inventory. A full list of adservers can be retrieved from the ad server service. Example: `"adservers":[{"id":11}, {"id":12}]`		

Examples

>> View a specific third-party pixel

In this example, we view a third-party pixel with ID 123.

```
$ curl -b cookies -X GET 'https://api.appnexus.com/thirdparty-pixel?id=123'

{
 "thirdparty-pixel":{
  "id":123,
  "active":true,
  "member_id":456,
  "advertiser_id": 789,
  "format":"raw-js",
  "content":"var img = new Image(); img.src="http://url.com/event/js?self=" + data;"
  "url":null,
  "secure_url":null,
  "members":null,
  "advertisers":[{"id":789}],
  "creatives":null
  "audit_status":"unaudited" // admin only
  "supply_exceptions":[
   {"name":"AdX"},
  ]
  "adservers":[
   {"id":123}
  ]
  }
}
```

>> Add a new third-party pixel

In this example, we create a new sell-side third-party pixel with ID 123. Note that parent object information is "null" because the JSON file we passed into the API did not specify a parent object for the pixel.

```
$ cat thirdparty-pixel.json
{
  "thirdparty-pixel":
    {
      "format":"url-html",
      "secure_url":"https://secureurl.com"
    }
}

$ curl -b cookies -c cookies -X POST -d @thirdparty_pixel
'https://api.appnexus.com/thirdparty-pixel'

{
    "response":{
        "status":"OK",
        "count":1,
        "id":123,
        "start_element":0,
        "num_elements":100,
        "thirdparty-pixel":{
            "id":123,
            "active":true,
            "name":null,
   "member_id":456,
            "advertiser_id":789,
            "publisher_id":null,
            "format":"url-html",
            "audit_status":"pending",
            "created_on":"2014-11-05 19:51:44",
            "last_modified":"2014-11-05 19:51:44",
            "url":null,
            "secure_url":"https://secureurl.com",
            "members":null,
            "advertisers":null,
            "publishers":null,
            "creatives":null,
            "supply_exceptions":null,
            "adservers":null

    }
}
```

>> Update a third-party pixel

In this example, we update a third-party pixel with the ID 123, applying it to all of the owning advertiser's creatives.

```
$ cat pixelupdate.json
{
  "thirdparty-pixel":
    {
      "advertisers":[{"id":789}]
    }
}

$ curl -b cookies -c cookies -X PUT -d @pixelupdate
'https://api.appnexus.com/thirdparty-pixel?id=123'

{
   "response":{
      "status":"OK",
      "id":123
   }
}
```

>> Delete a third-party pixel

In this example, we delete a third-party pixel with the ID 123.

```
$ curl -b cookies -X DELETE 'https://api.appnexus.com/thirdparty-pixel?id=123'

{
   "response":{
      "status":"OK"
   }
}
```

Related Topics

- API Semantics
- API Best Practices
- Creative Service
- Advertiser Service